2017

A Novel Flow-Aware Fair Scheduler for Multi Transmit/Receive Wireless Networks

Luyao Wang
University of Wollongong, lw233@uowmail.edu.au

Kwan-Wu Chin
University of Wollongong, kwanwu@uow.edu.au

Sieteng Soh
Curtin University of Technology, Curtin University, s.soh@curtin.edu.au

Tengjiao He
University of Wollongong, th877@uowmail.edu.au

A Novel Flow-Aware Fair Scheduler for Multi Transmit/Receive Wireless Networks

Abstract

This paper considers the problem of deriving a time-division multiple-access (TDMA) schedule for multi-hop wireless networks that allow nodes to perform multiple transmissions/receptions to/from all of their neighbors simultaneously over the same frequency. To date, there are a number of link schedulers for the networks in question but they do not consider flow rate or any notion of fairness when deriving a TDMA schedule. Henceforth, we address this critical gap by proposing a link scheduler, called Algo-Fair, that, in addition to maximizing the number of links in each slot, also provides fairness among flows. In addition, it uses a novel augmentation step to distribute spare capacity fairly among flows. We believe this step is general and can be applied readily in other forms of wireless networks. Apart from that, Algo-Fair generates a schedule directly while yielding a fair rate allocation. This is different from existing methods that first use a flow contention graph to compute a fair share before deriving the corresponding schedule, which may not exist. Numerical results show Algo-Fair has higher fairness, minimum rate, and average total throughput than competing approaches, and low end-to-end delay.

Disciplines

Engineering | Science and Technology Studies

Publication Details

L. Wang, K. Chin, S. Soh & T. He, "A Novel Flow-Aware Fair Scheduler for Multi Transmit/Receive Wireless Networks," IEEE Access, vol. 5, pp. 10456-10468, 2017.

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 20, 2017, accepted May 19, 2017, date of publication May 29, 2017, date of current version June 27, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2708160

A Novel Flow-Aware Fair Scheduler for Multi Transmit/Receive Wireless Networks

LUYAO WANG[1], KWAN-WU CHIN[1], SIETENG SOH[2], (Member, IEEE), AND TENGJIAO HE[1]
[1]School of Electrical, Computer, and Telecommunications Engineering, University of Wollongong, Wollongong NSW 2522, Australia
[2]Department of Computing, Curtin University, Bentley, WA 6102, Australia

Corresponding author: Kwan-Wu Chin (kwanwu@uow.edu.au)

ABSTRACT This paper considers the problem of deriving a time-division multiple-access (TDMA) schedule for multi-hop wireless networks that allow nodes to perform multiple transmissions/receptions to/from all of their neighbors simultaneously over the *same* frequency. To date, there are a number of link schedulers for the networks in question but they do not consider flow rate or any notion of fairness when deriving a TDMA schedule. Henceforth, we address this critical gap by proposing a link scheduler, called Algo-Fair, that, in addition to maximizing the number of links in each slot, also provides fairness among flows. In addition, it uses a novel augmentation step to distribute spare capacity fairly among flows. We believe this step is general and can be applied readily in other forms of wireless networks. Apart from that, Algo-Fair generates a schedule directly while yielding a fair rate allocation. This is different from existing methods that first use a flow contention graph to compute a fair share before deriving the corresponding schedule, which may not exist. Numerical results show Algo-Fair has higher fairness, minimum rate, and average total throughput than competing approaches, and low end-to-end delay.

INDEX TERMS Time division multiple access, wireless mesh networks, scheduling algorithms.

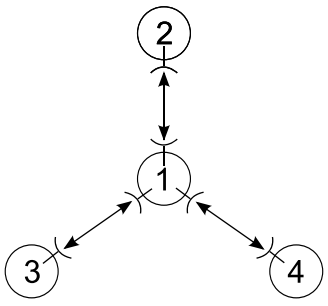
I. INTRODUCTION

Wireless Mesh Networks (WMNs) can be used to improve the coverage, capacity and reliability of an existing communication infrastructure. To this end, the IEEE has standardized two meshing technologies. The first is IEEE 802.11s [7], which extends conventional Access Points (APs) with meshing capabilities, aka Mesh Points (MPs). These mesh points can then form a multi-hop wireless network that expands the coverage of an existing Wireless Local Area Network (WLAN). The second is IEEE 802.16, aka WiMAX [8]. This standard supports two transmission modes: Point-to-Multipoint (PMP) and mesh. In PMP, traffic is transmitted between a Base Station (BS) and Subscriber Stations (SSs). However, in mesh mode, traffic can also be relayed between SSs. Apart from these standards, mesh technology will also play a crucial role in future IEEE 802.11ad and 5G networks, where the high path loss on the 60 GHz channel will require routers to have relaying capability [13] [24]. In particular, in the foregone examples, routers, equivalently MPs/SSs, are required to relay flows. Here, we define a *flow* as the traffic between two routers; that is, the aggregated traffic from clients associated to a router that is destined for clients connected to another router.

In order to ensure high flow rates, network capacity must be high. In this respect, researchers have considered equipping nodes/routers with multiple transmit (Tx) or receive (Rx) or MTR capability [5]. This means routers are able to transmit/receive N packets to/from N distinct neighbors simultaneously. Consequently, unlike existing WMNs that assume a *single* transmission at a given time within a given range, MTR WMNs have a much higher capacity. To date, MTR WMNs can be realized using three different wireless systems. First, Raman and Chebrolu [19] used a WMN to interconnect rural villages in India. Wireless routers have a number of IEEE 802.11 radios, each connected to a high-gain parabolic antenna. To ensure correct reception, they fixed the transmission power and ensure incident links are apart by at least 30°. It is worth pointing out that all routers operate over a single frequency because of other devices operating in the 2.5 GHz band. The second method to realize a MTR system is to equip nodes with 60 GHz radios. Consequently, nodes have high directivity [13]. Interestingly, Mudumbai *et al.* [13] show that the interference between neighboring links can be ignored, meaning there is no secondary interference. In fact, mm-wave or 60 GHz wireless links can be treated as *pseudo-wires*. The final method to

achieve MTR is to take advantage of advances in Multiple Input Multiple Output (MIMO) communications. Specifically, nodes have multiple antenna elements, which they then use to null/suppress interference, and also form data links to their neighbors over the same frequency band. In addition, these nodes can weight their transmission power according to channel state information. For example, Chu and Wang [6] demonstrated a system where a node transmits concurrent streams to multiple neighbors. Receiver nodes can then use minimum mean square error sequential interference cancellation (MMSE-SIC) to decode multiple streams. It is worthwhile noting that in the aforementioned MTR systems, all nodes are *half-duplex*, meaning a node cannot transmit and receive at the same time. This is because the self-interference caused by a node's transmissions overwhelm any signals it is trying to receive. From here on, we term the half-duplex nature of nodes as the no *mix-tx-rx* constraint. We remark that our work is different from those that consider multi-radio multi-channel (MR-MC) [23]. This is because in MTR WMNs, all nodes operate on the *same* frequency.

Link scheduling is a key problem in MTR WMNs. In particular, a link scheduler is required to ensure all transmitting links do not experience collision and nodes receive their required transmission opportunities. To this end, we consider a Time Division Multiple Access (TDMA) based scheduler that divides time into slots, and the links scheduled in the same time slot are collision-free. The total number of slots used to satisfy all transmission demands or afford each link at least one transmission opportunity is called a TDMA schedule or superframe. Figure 1 shows a simple TDMA-based MTR WMN and its schedule. We see that node 1 is able to transmit/receive to/from three neighbors in slots 1 and 2, respectively. This superframe contains two slots and is repeated over time. From this example, we see that a short superframe is critical because it ensures links are activated frequently and maximizes the number of links in each slot. In turn, this ensures a high network capacity.



Slot 1	Slot 2
1 → 2	2 → 1
1 → 3	3 → 1
1 → 4	4 → 1

FIGURE 1. An example MTR wireless network along with its TDMA schedule.

To date, the main *aim* of existing single channel TDMA-based MTR WMN schedulers is to derive a conflict-free schedule that minimizes the superframe length or maximizes the number of links in each slot whilst allowing each link at least one transmission opportunity or satisfying the given demands. They, however, do not consider whether the rate allocated to flows is fair and thus, they may starve flows. Specifically, flows over multiple hops may receive less bandwidth than single-hop flows. The rate allocated to a multi-hop flow is decided by the minimum number of slots assigned to the links on its path. Consider the topology in Figure 2. It has a multi-hop flow f_1 and a single hop flow f_2. A possible link schedule that maximizes the number of links in each slot is shown in Table 1. The rate allocated to f_1 is $1/3$ because the superframe length is 3 and link BC and CD are assigned with one slot respectively. However, the rate of f_2 is $2/3$ because link DC is active in two slots. In this case, the multi-hop flow is starved. Another possible link schedule is shown in Table 2 that allocates $1/2$ capacity to f_1 and f_2 respectively. In this case, the two flows share the channel capacity fairly. Thus, a challenging problem is how to allocate bandwidth to flows fairly as well as maximize network capacity.

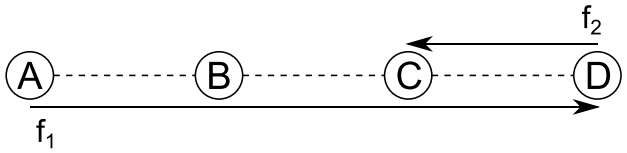


FIGURE 2. An example topology with two flows: $f_1 : A - B - C - D$ and $f_2 : D - C$.

TABLE 1. An example superframe that allocates $1/3$ capacity to f_1 and $2/3$ capacity to f_2.

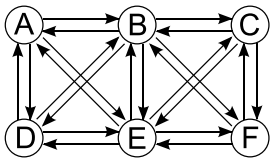
Slot 1	Slot 2	Slot 3
AB	BC	CD
DC	DC	AB

TABLE 2. An example superframe that allocates $1/2$ capacity to f_1 and f_2 respectively.

Slot 1	Slot 2
AB	BC
CD	DC

Given the importance of the foregone problem and the lack of solutions, see Section II, in this paper, we make the first step towards a TDMA-based link scheduler that generates a superframe that ensures end-to-end fairness while also maximizing the number of links in each slot. Before proceeding further, we will first need to define three key concepts: opportunistic links, opportunistic slots, and opportunistic flows.

Consider the schedule in Figure 3 in which every link can be considered as a single-hop flow and is activated at least once. Observe that all transmissions in each slot adhere to the no *mix-tx-rx* constraint. Also, links AE, BD, CE and BF that have been activated in slots prior to slot 3 do not conflict with links AD, BE and CF that are currently allocated to slot-3. To maximize network capacity, these non-conflicting links can also be activated in slot 3. With this example in



Slot 1	Slot 2	Slot 3	Slot 4
BA	AB	AD	DA
BD	DB	BE	EB
BC	CB	CF	FC
BF	FB	AE*	DB*
EA	AE	BD*	EA*
ED	DE	BF*	EC*
EC	CE	CE*	FB*
EF	FE		

FIGURE 3. A '2-boxes' topology and its TDMA MTR schedule/superframe. Opportunistic links are mark with an asterisk. A time slot containing an opportunistic link is called an opportunistic slot. A flow in which each link on its path can be activated in an opportunistic slot is called an opportunistic flow.

hand, we define *opportunistic links* as the links that can be added into a slot without causing a conflict with other links that are already allocated to the slot; see links with an asterisk. Observe that opportunistic links have a higher number of allocated slots, meaning they have a higher capacity. We call a slot with at least one opportunistic link an *opportunistic slot*; e.g., slot 3 and 4. Finally, we define a flow as an *opportunistic flow* if *all* links on its path are opportunistic links. As an example, a two-hop flow from node A to node C via node E is an opportunistic flow because its AE and EC are opportunistic links in slot 3 and 4.

We are now ready to list our *contributions*: (1) we propose a MTR scheduler named Algo-Fair that is fundamentally different from prior works that consider only one transmission/reception at a given time. In fact, to the best of our knowledge, our work is the first that jointly addresses the problem of maximizing the number of links in each slot and ensuring additional capacity is allocated fairly amongst end-to-end flows in MTR WMNs, (2) Algo-Fair identifies opportunistic flows, and uses a novel augmentation step that allows opportunistic slots to be shared fairly amongst flows. We believe this augmentation step is general and can be applied to other non-MTR wireless networks that seek to derive a TDMA superframe whilst ensuring all flows have a fair rate, (3) we are the first to use a scheduling approach to approximate Max-Min Fairness (MMF). Compared to other MMF related works, we directly generate a superframe that yields a fair rate allocation without having to use a flow contention graph. This is significant as the fair rate computed by these existing works may not have a corresponding schedule [9]. Numerical results show that Algo-Fair is able to achieve the optimal MMF allocation in some cases, and on average, has a gap of 3.7% to the optimal solution. Moreover, Algo-Fair is able to generate a higher minimum flow rate as well as a higher average throughput as compared to other algorithms, while producing the second best low end-to-end delay.

Next, in Section II, we present relevant works and place our work with respect to those that consider multiple channels and radios [10], or works that assume the physical or protocol interference model. Our network model and key definitions are introduced in Section III. In Section IV we present the key features of our link scheduler called Algo-Fair. A key exposition is its augmentation step, whereby the superframe in the previous iteration is replicated to free up opportunistic slots that are then assigned to flows fairly. The details of our scheduler are presented in Section V. We also provide a worked example. The key properties of Algo-Fair are outlined in Section VI. Our research methodology is presented in Section VII. Our conclusion and possible future works are presented in Section VIII.

II. RELATED WORKS

To the best of our knowledge, there are no schedulers that aim to derive a superframe that consider the throughput and fairness of end-to-end flows. Critically, they do not consider networks with MTR capability and do not incorporate an *augmentation* step to exploit opportunistic slots and assign these slots fairly. In addition, our work is the first to use a superframe approach to approximate max-min fairness (MMF). Compared to other MMF related works, our work generates a superframe that yields a fair allocation directly without having to use a flow contention graph. This is significant as existing works may yield a fair share in which there is no link schedule [9]. We will elaborate on these points next.

To date, there are only a few link schedulers that are designed for MTR WMNs. Raman and Chebrolu [19] proposed a Spatial reuse Time Division Multiple Access (STDMA) protocol, named 2P, to generate a schedule whereby nodes that have transmitted in slot i will become receivers in slot $i+1$. However, 2P assumes a bipartite topology. Subsequently, Chin *et al.* [5] propose Algo-1, a scheduler that operates over an arbitrary topology and solves the NP-complete MAX-CUT problem in every slot pair using a polynomial time heuristic. However, similar to 2P, Algo-1 generates a new schedule in every other slots, whereby transmitting nodes in slot i automatically become receivers in slot $i+1$. This limitation is addressed in [11] where the authors proposed Algo-2, a scheduler that generates shorter superframe lengths than Algo-1, and does so by generating a new schedule in every slot as opposed to every other slot, and also assigns a higher priority to links with a high load. In [4], [6], and [14], the authors aim to maximize the capacity of MIMO-based WMNs. The approach in [4] uses a Mixed Integer Linear Program (MILP) to maximize the total flow rate whereby the decision variables are the set of antennas to be activated in each slot and link data rate. Reference [6] aims to maximize network capacity by constructing an optimal sub-graph that includes nodes and antennas with a higher priority in each time slot. Mumey *et al.* [14] propose a heuristic algorithm that constructs a contention graph and uses graph coloring to separate links that cannot be activated simultaneously because of half-duplex and interference constraint. Notably, Chu and Wang [6] and Mumey *et al.* [14] did not consider end-to-end flow rate and fairness. In addition, it is unclear how the aforementioned approaches can be modified to take advantage of opportunistic slots, and share this additional capacity fairly amongst flows. Qiao *et al.* [18] and

Niu *et al.* [15] aim to maximize the capacity of 60 GHz WMNs. Both works exploit spatial reuse and the utilization of highly directional antenna. However, both works do not consider allocating capacity to flows fairly.

We note that the fairness of end-to-end flows has been considered for many other systems, e.g., Carrier Sense Multiple Access (CSMA) based WMNs, and those with multi-channel multi-radios; see [23]. Fundamentally, these systems are different as they assume a different interference model. As illustrated in Section I, a MTR capable node can transmit/receive multiple packets simultaneously. This means a node does not experience collision when two or more neighbors transmit to it. Apart from that, a large body of works aim to match nodes and assume paired nodes or links in a matching do not interfere. For example, in [26], the proposed algorithm constructs a matching in each slot and uses tokens to track the fair rate of flows. Penttinen *et al.* [16] propose a fair scheduler that constructs matchings in each slot according to flow priority. Both [26] and [16] only consider single-hop flows. Note, we consider flows that span multiple hops. In this respect, works such as [21] extended the token idea in [26] to multi-hop flows. However, [16], [26], and [21] did not generate a superframe, which is different to our objective. Moreover, their link scheduler derives the maximal matchings in each slot whereas ours involve solving the MAX CUT problem in each slot.

A popular approach to determine fair end-to-end flow rates is to formulate the problem as a mathematical program. The authors of [1] proposed an Integer Linear Programming (ILP) to maximize the minimum flow rate in a tree-based OFDMA-based WiMAX network while considering interference, single transceiver and flow constraints. Pioro *et al.* [17] proposed several MILP formulations, and also an iterative solution that maximizes each flow while retaining previously computed flow rates. These approaches, however, are computationally expensive. Moreover, existing MILP formulations assume flows can be routed over multiple paths. However, in this paper, we consider each flow is routed over one path. This is important to TCP because out-of-order packets degrade its performance. We note that formulating a joint routing and link scheduling optimization model for MTR WMNs that yields a fair rate allocation is an interesting future research direction.

Apart from that, there are works that employ a flow contention graph to determine the fair share of each flow. Wang *et al.* [31] adapt the notion of MMF to when allocating channel time. They first construct a flow contention graph, and then allocate equal channel time to each multi-hop flow. In [17], besides the MILP, the authors also proposed a heuristic water-filling algorithm based on a flow contention graph to solve large problem instances. However, building a flow contention graph is computationally expensive because it involves checking all link pairs. Besides that, a number of works employ a heuristic algorithm to allocate flow rates. For example, Wang and Jia [29] consider varying link capacities, and propose a centralized algorithm that iterates through all nodes and calculates the said fair fraction for all flows that traverse each node. The resulting fair share of each link is then used to calculate the number of time slots to be assigned to each flow. In this work, the authors assume each node only transmits/receives to/from one neighbor in each time slot, which is different to our MTR system. In the aforementioned works, a fair share for each flow needs to be calculated before scheduling links. In contrast, our approach generates a superframe directly whilst also ensuring flows receive a fair share of the available capacity.

Lastly, we provide a brief discussion on distributed fair schedulers. Note, this is beyond the scope of this paper. To date, there are only a few distributed link schedulers that are designed for MTR WMNs. Wang *et al.* [28] proposed a distributed version of Algo-2 [11], but did not consider end-to-end flow rate and fairness. Penttinen *et al.* [6] proposed a distributed link scheduler for MIMO-based WMNs where a node first decides whether it is a transmitter. If so, it activates high priority streams. Another distributed MTR link scheduler is proposed in 33 [33], where the authors employ the celebrated backpressure algorithm [25] to achieve proportional fairness. However, both [6] and [33] do not aim to generate a superframe nor take advantage of opportunistic slots. We note that developing a MTR aware distributed algorithm that derives a superframe that maximizes the number of links in each slot as well as exploits opportunistic slots via a novel augmentation step is an interesting research direction.

In summary, our work is distinct from prior works given that it considers end-to-end fairness for TDMA-based wireless networks with MTR capability. Advantageously, our approach does not rely on a flow contention graph or a LP solver; both of which are computationally expensive and may yield a fair flow rate in which there is no corresponding schedule. We, on the other hand, consider both fairness and superframe generation jointly. A key highlight, as detailed in Section IV, is a general augmentation operation that allows opportunistic slots to be assigned to links fairly. In the next section, we will introduce our network model and key definitions.

III. PRELIMINARIES

We consider an arbitrary, single channel, TDMA-based MTR WMN $G(V, E)$, where V is the set of nodes, and E contains directed links; each link l has capacity C_l. We assume all links have the same capacity. Further, time is divided into time slots, and each slot is sufficient to transmit a single packet. A superframe S is defined as a grouping of slots; hence, the superframe length $|S|$ is equal to the number of slots. Note, as TDMA requires nodes to be synchronized, we assume nodes have a GPS unit; one that has a resolution of 0.5 microseconds can be found in [12]. Let $\mathcal{F}$ represent a set of flows. Each flow $f_i \in \mathcal{F}$, indexed by $i \in [1, |\mathcal{F}|]$, is routed over the shortest path, in terms of hops. With a slight abuse of notation, we will also use f_i to indicate the set of links on the path. We assume flows are greedy, meaning they will consume any given bandwidth. We use r_i to denote the

portion of bandwidth allocated to flow f_i, and r is a vector representing the rate allocated to all flows in $\mathcal{F}$; i.e., $r = (r_1, r_2, \ldots, r_{|\mathcal{F}|})$. For example, $r_i = 1/3$ means all links on the route for f_i dedicate $1/3$ of their capacity to the flow. Further, a link l is defined as flow f_i's *bottleneck link* if and only if link l's capacity is fully utilized by all flows crossing it and the rate of f_i is higher than or equal to the rate of other flows crossing l. We then have the following definition,

Definition 1: A rate allocation for $\mathcal{F}$ is feasible if it does not exceed the capacity C_l of each link l.

We say a schedule is *feasible* if it satisfies the following definition.

Definition 2: A link schedule for a MTR WMN is feasible if all transmissions in each slot satisfy the no mix-tx-rx *constraint.*

As we will elaborate in Section IV, our approach employs Algo-2 [11], a centralized MTR link scheduler. Critically, as reported in [11], Algo-2 is able to produce a TDMA schedule that maximizes capacity with a short superframe length. To aid exposition later on, we briefly highlight the key steps performed by Algo-2. Its goal is to generate the maximal set of links in each slot. This is achieved using a heuristic to the well-known, NP-complete, MAXCUT problem [27]. Specifically, it creates a maximal bipartite graph in each slot and derives a minimal superframe length that ensures all link demands are satisfied. In each slot k, Algo-2 creates two node sets: Set1 and Set2. The nodes in Set2 will transmit to Set1 in slot k. The two sets are constructed as follows. Initially, Algo-2 includes all nodes into Set1 and sets Set2 to empty. It then selects the node with the highest δ value in Set1 and moves it into Set2, where the δ value of a node A is the difference between the total weight of links from A to other nodes in Set1 and the sum of link weights from nodes in Set2 to node A. It then repeats the second step until the δ value of all nodes in Set1 is less than or equal to zero. After generating one slot, the weight of activated links is reduced by one. The algorithm terminates when the weight of all links is zero. We remark that the weight of a link l corresponds to the number of slots assigned to it in the superframe generated by Algo-2. Note, Algo-2 can be replaced with any future schedulers with better performance. We note that our algorithm can also be used by other TDMA schedulers for non MTR systems. This is a subject of an immediate future work.

We now use the '2-boxes' topology and the first scheduled slot shown in Figure 3 to show how Algo-2 works. We assume the weight of each link is one. At the beginning, Algo-2 creates two node sets: Set1= A, B, C, D, E, F and Set2= $\emptyset$. The δ value of each node is $\{\delta_A, \delta_B, \delta_C, \delta_D, \delta_E, \delta_F\} = \{3, 5, 3, 3, 5, 3\}$. Then, Algo-2 moves node B to Set2 and updates the δ value of other nodes to $\{\delta_A, \delta_C, \delta_D, \delta_E, \delta_F\} = \{1, 1, 1, 3, 1\}$. Node D is then moved to Set2 because δ_D is the biggest. The new δ values are $\{\delta_A, \delta_C, \delta_D, \delta_F\} = \{-1, -1, -1, -1\}$. Also-2 then stops moving nodes because all δ values are negative. In the first slot, the nodes in Set2= $\{B, E\}$ will transmit to nodes in Set1= $\{A, C, D, F\}$.

As mentioned earlier, we consider end-to-end flow fairness. A well-known fairness criterion is MMF, which has the following standard definition [3]:

Definition 3: A rate allocation vector $r = (r_1, r_2, \ldots, r_{|\mathcal{F}|})$ is a MMF rate allocation for flows in $\mathcal{F}$ if, with respect to any other rate allocation vector $r' = (r'_1, r'_2, \ldots, r'_{|\mathcal{F}|})$, there exist a flow $f_i \in \mathcal{F}$ with $r_i < r'_i$, and another flow $f_j \in \mathcal{F}$ with $r_j \leq r_i$ and $r_j > r'_j$.

Note that each flow in $\mathcal{F}$ has at least one bottleneck link, and the rate of each flow in a feasible MMF rate allocation cannot be increased without decreasing the rate of any other flows having less or equal rates [3]. We will use MMF as the main fairness criterion to optimize, although other fairness metric can be used as well. Given this definition, we then say,

Definition 4: A feasible link schedule for a MTR WMN satisfies MMF if the rate allocation for all flows in the set $\mathcal{F}$ satisfies Definition 1, 2 and 3.

We are now ready to outline the problem. Our aim is to derive a link schedule for MTR WMNs that yields a fair rate allocation to end-to-end flows. Formally, let S_n denote the set of links activated in slot $n \in [1, |S|]$, and $\mathcal{F}_l$ is the set of flows traversing link l. Thus, given a MTR WMN modeled as $G(V, E)$, a set of flows $\mathcal{F}$, the fair link scheduling problem is to find a feasible link schedule, i.e., $S = \{S_1, S_2, \ldots, S_{|S|}\}$, where (1) the rate allocation $r = (r_1, r_2, \ldots, r_{|\mathcal{F}|})$ is feasible, i.e., for any link l in E, $\sum_{i \in \mathcal{F}_l} r_i \leq C_l$, and (2) the generated r is a MMF rate allocation.

We remark that generating a rate allocation that satisfies MMF for wireless networks is more complex than wired networks because of interference between transmitting/receiving links. Also, the resulting solution is an *approximation* because the superframe generated by Algo-2 is not optimal and hence, the capacity is sub-optimal. However, as discussed in Section IV and V, our proposed algorithm is able to ensure a high minimum flow rate.

IV. ALGO-FAIR – KEY IDEAS

Algo-Fair first generates a *basic* superframe, using Algo-2 [11], where each flow is assigned a weight of one. This means each links of a flow f_i will be assigned one slot in the basic superframe by Algo-2, and thus Algo-Fair generates a schedule in which each link is activated in at least one slot. In the next step, Algo-Fair aims to identify opportunistic flows. A key problem in this step is that some conflicting links may have the same opportunistic slot. Consequently, they cannot be activated in the same slot. To this end, our algorithm employs a superframe augmentation step that duplicates the superframe in the prior iteration such that all conflicting links can be activated opportunistically. More importantly, previously scheduled links and thus the flow rate of non-opportunistic flows are unaffected. Our algorithm terminates when the rate increase of opportunistic flows is less than or equal to a parameter called ϵ; we will elaborate on this parameter in Section VI.

Algo-Fair creates the desired superframe iteratively, starting with the basic superframe, denoted as B_0. Figure 4 shows

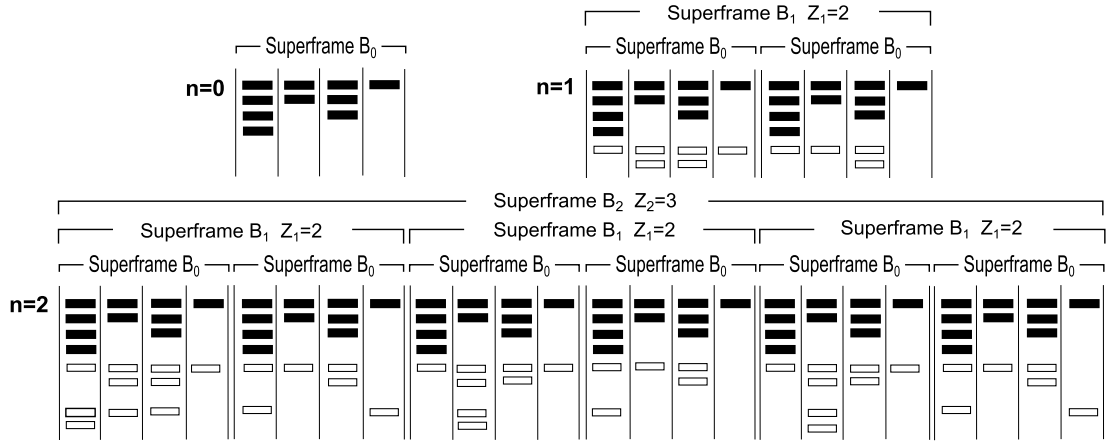


FIGURE 4. Example superframes created by our algorithm.

an example superframe B_n for iteration n; the *black* bars indicate the links scheduled in the superframe. Observe that superframe B_1 is constructed from B_0, and B_2 from B_1; observe also the links scheduled in B_{i-1} retain their period in B_i. Let Z_n be the number of B_{n-1} copies used to construct B_n in the n-th iteration, where $Z_n \geq 1$. For example, in iteration $n = 1$, we have $Z_1 = 2$.

We will now explain the key superframe construction or *augmentation* operation performed by Algo-Fair in iteration $n > 0$. The operation, denoted by $\bigoplus$, is defined as follows,

Definition 5: The $\bigoplus$ operator on any two superframes B_i and B_j, i.e., $B_i \bigoplus B_j$, appends the slots in B_j, and thus all links scheduled in B_j, into B_i starting at slot $|B_i|+1$. Alternatively, $B_k = B_i \bigoplus B_j$ results in a B_k such that its first $|B_i|$ slots are exactly the same as B_i and its last $|B_j|$ slots are exactly those of B_j.

Note that the $\bigoplus$ operator is non-commutative, i.e., $B_i \bigoplus B_j \neq B_j \bigoplus B_i$. For example, in Figure 4, superframe B_1 is generated via the operation $B_1 = B_0 \bigoplus B_0$. The new superframe length is the total number of slots in the combined superframes. Observe that this operation preserves the flow rate of allocated flows. That is, if a link is activated in slot k of superframe B_0, the link is activated with the same period in subsequent superframes.

A very important observation is as follows. Consider an opportunistic slot k in B_0, and also two conflicting links l_1 and l_2 that can be scheduled opportunistically, but not simultaneously, in slot k. In superframe B_1, if there is only one copy of B_0, then either l_1 or l_2 can be scheduled in slot k. Now, if we augment B_1 with another copy of B_0, then l_1 and l_2 can be scheduled in the first and second copy of B_0, respectively; we say these links/flows have received *additional bandwidth*. We will denote the additional bandwidth received by all flows in iteration n as Δ_n. We will now define opportunistic flows formally.

Definition 6: A flow in iteration n is an opportunistic flow if its links can be activated in some slot(s) in superframe B_{n-1}.

As we shall see in Section V, at the beginning of iteration n, Algo-Fair determines whether there are any flows where all their links can be activated opportunistically in the superframe generated in iteration $n-1$. If so, the flows are marked as opportunistic flows in iteration n. In order to keep track of opportunistic flows, we define two sets, $F \subseteq \mathcal{F}$ and $F' \subseteq F$, which contain the opportunistic flows in superframe B_{n-1} and B_n respectively. In addition, we will use the multiset $\mathcal{L}$ to record all links used by flows in F'. Note that $\mathcal{L}$ may contain duplicated links, corresponding to links used by multiple flows in F'.

V. ALGORITHM DETAILS

We are now ready to delve into the details; see Algorithm 1. Algo-Fair takes as input a graph $G(V, E)$, a set of flows $\mathcal{F}$ and ϵ. Algo-Fair sets the flag m to one when it finds a flow f_i whereby all its links can be activated opportunistically in superframe B_{n-1}. Let L be a subset of $\mathcal{L}$; i.e., $L \subseteq \mathcal{L}$, that contains all opportunistic links in slot k. Note that the links in L may conflict with each other. Hence, we use a function called MAXCUT() to determine a maximal set of non-interfering links. Briefly, MAXCUT() divides nodes into two sets such that the total number of links between nodes in one set to nodes in the other is maximal. It outputs the links between these two sets. We define S as the final superframe returned by our algorithm.

Algo-Fair starts by assigning each flow in $\mathcal{F}$ a weight of one. Using Algo-2 [11], it generates superframe B_0 from the graph G; see *line 1-2* of Algorithm 1. Then the rate allocated to all flows in $\mathcal{F}$ is initialized to $1/|B_0|$. Recall that $|B_0|$ is the superframe length and the links of each flow is activated once in the superframe; see *line 3*. The set F is initialized to $\mathcal{F}$. Algo-Fair sets n and Δ_n to zero, and the set F' is initialized to empty; see *line 4*. In each iteration n, it includes into set F' those flows in which all their links can be opportunistic links in B_{n-1}. In addition, it includes all their links into the multiset $\mathcal{L}$; see *line 7-16*. If no flows in F can be included into F', Algo-Fair returns the superframe generated in the

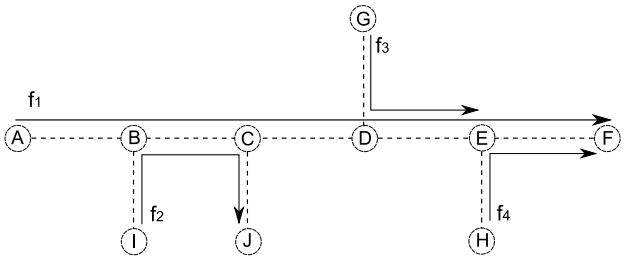


FIGURE 5. An MMF example.

TABLE 3. Basic superframe generated by Algo-2.

Slot 1	Slot 2	Slot 3	Slot 4
BC	EF	BC	EF
DE	GD	DE	
HE	CD		
	CJ		
	AB		
	IB		

previous iteration; see *line 17-18, 44*. If set F' is not empty, the value of B_n is initialized to B_{n-1}, and the value of k and Z_n is set to one; see *line 20*. Let l indicate a link in set $\mathcal{L}$. In each slot k, if link l is an opportunistic link, it is included into the set L. The next step is to determine which links in L can be active simultaneously. To do this, it generates the MAX-CUT from the set L and schedules the returned links in slot k as opportunistic links, and removes the links from $\mathcal{L}$; see *line 26*. When all links in $\mathcal{L}$ cannot be added in the given $|B_n|$ slots, in *line 30*, Algo-Fair will augment B_n with B_{n-1}. This step is repeated until all links in $\mathcal{L}$ are scheduled. The number of B_{n-1} copies used in iteration n is recorded in the variable Z_n; see *line 21-33* of Algorithm 1. Recall that Δ_n is the additional bandwidth generated in each iteration. Thus the value of Δ_n is updated to $1/|B_n|$ because each flow in set F' receives one more activation in the new superframe B_n; see *line 34*. If $0 < \Delta_n < \epsilon$, the algorithm will break and return the superframe generated in the last iteration; see *line 5, 41*. At the end of each iteration n, Algo-Fair increases the allocated bandwidth of each flow in F' by Δ_n; see *line 35-37*. Then the set F is updated to F' and F' is reset to empty; see *line 38*. The output of Algo-Fair is superframe S and vector r.

We now show how Algo-Fair computes the flow rates for the example shown in Figure 5.

- *Input:* It takes as input $G(V, E)$, the set of flows $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$ and $\epsilon = 0.001$.
- *Output:* It outputs a superframe S and a vector r that represents the rate allocated to all flows in $\mathcal{F}$, i.e., $r = (r_1, r_2, r_3, r_4)$.
- *Line 1-4:* It assigns each flow a weight of one and generates superframe B_0 using Algo-2; the results are shown in Table 3. The superframe length is $|B_0| = 4$. As flows receive one slot, each flow has a rate of $1/|B_0| = 1/4$; i.e., $r_1 = r_2 = r_3 = r_4 = 1/4$. The set F is initialized to $\mathcal{F}$, i.e., $F = \{f_1, f_2, f_3, f_4\}$. It also sets the value of n and Δ_n to zero and set F' to empty.

TABLE 4. Final superframe generated by Algo-Fair.

Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8
BC	EF	BC	EF	BC	EF	BC	EF
DE	GD	DE	$IB*$	DE	GD	DE	$BC*$
HE	CD		$CJ*$	HE	CD		
	CJ				CJ		
	AB				AB		
	IB				IB		

- *Line 7-16:* In iteration $n = 1$, link DE and EF cannot be opportunistic links in any slot in B_0, thus flow f_1, f_3 and f_4 will not be added into the set F'. On the other hand, all three links IB, BC and CJ of flow f_2 can be opportunistic links in slot 4. Thus, $F' = \{f_2\}$ and $\mathcal{L} = \{IB, BC, CJ\}$.
- *Line 17-33:* It does not return B_0 at *line 17-18* because the set F' is not empty. It then initializes the value of k and Z_n to one and $B_1 = B_0$. When $k = 4$, set L is not empty; i.e., $L = \{IB, BC, CJ\}$. To determine which links in L can be added to slot k, the function MAXCUT() is called and in this case returns two links: IB and CJ. These links are added into slot 4 and removed from set $\mathcal{L}$. Thus the only link left in $\mathcal{L}$ is BC. When the value of k is higher than $|B_1| = 4$, the algorithm augments B_1 by performing $B_1 = B_1 \bigoplus B_0$, and increases the value of Z_1 to two. Consequently, the superframe length of B_1 is 8 slots. When the value of k is 8, link BC is scheduled in slot 8 and set $\mathcal{L}$ is empty.
- *Line 34-41:* The value of Δ_n is updated to $1/|B_1| = 1/8$. Flow f_2 is the only flow in set F', thus the rate allocated to f_2, r_2 is increased by $\Delta_n = 1/8$. The total rate allocated to flow f_2 is $3/8$ at the end of the first iteration. The algorithm then copies the set F' to F and set F' to empty. In the second iteration, flow f_2 is the only flow in set F. The set F' is empty after testing flow f_2 because all three links cannot be opportunistic links in B_1.
  Finally, it returns $S = B_1$ as well as $r = (r_1, r_2, r_3, r_4)$, where $r_1 = 1/4$, $r_2 = 3/8$, $r_3 = 1/4$ and $r_4 = 1/4$.

To conclude, we briefly comment on the implementation of Algo-Fair. In a WMN, there is usually one or more gateways. A gateway can then be made responsible for processing flow requests; e.g., the gateway might simply be a software defined network controller [2]. Then using Algo-Fair, a new schedule can be computed and disseminated to routers. In terms of schedule dissemination, one possible implementation is to have two mini-slots at the beginning of each time slot. The first mini slot is used to receive the computed schedule from a parent node and the second mini slot is used to forward the schedule to all children.

VI. ANALYSIS

We now discuss several properties of Algo-Fair, and its time complexity. We prove that the superframe length in each iteration and the number of iterations are finite. Then we show that the resulting flow rates approximate MMF. Finally, we outline its running time complexity.

Algorithm 1 Algo-Fair

input : $G(V, E)$, $\mathcal{F}$, ϵ
output: link schedule S, rate allocation $r = (r_1, r_2, \ldots, r_{|\mathcal{F}|})$

1 Assign each flow $f_i \in \mathcal{F}$ a weight of one
2 $B_0 \leftarrow$ `Algo-2` (G)
3 $r_1 = r_2 = \cdots = r_{|\mathcal{F}|} \leftarrow 1/|B_0|$
4 $F \leftarrow \mathcal{F}$, $n \leftarrow 0$, $\Delta_n \leftarrow 0$, $F' \leftarrow \emptyset$
5 **while** $\Delta_n = 0$ *OR* $\Delta_n \geq \epsilon$ **do**
6 $n \leftarrow n + 1$
7 **for** $i \leftarrow 1$ **to** $|F|$ **do**
8 $m \leftarrow 1$ `// m is a flag`
9 **for** $l \in f_i$ **do**
10 **if** l *cannot be an opportunistic link in any slot* **then** $m \leftarrow 0$
11 **end**
12 **if** $m = 1$ **then**
13 Add flow f_i into F'
14 Add all links of f_i into $\mathcal{L}$
15 **end**
16 **end**
17 **if** $F' = \emptyset$ **then**
18 break
19 **else**
20 $k \leftarrow 1$, $Z_n \leftarrow 1$, $B_n \leftarrow B_{n-1}$, $L \leftarrow \emptyset$
21 **while** $\mathcal{L} \neq \emptyset$ **do**
22 **for** $l \in \mathcal{L}$ **do**
23 **if** l *can be an opportunistic link in slot* k **then** Add l into L
24 **end**
25 **if** $L \neq \emptyset$ **then**
26 Add `MAXCUT` (L) in slot k, $\mathcal{L} = \mathcal{L} \setminus$ `MAXCUT` (L)
27 **end**
28 $k \leftarrow k + 1$
29 **if** $k > |B_n|$ **then**
30 $B_n \leftarrow B_n \bigoplus B_{n-1}$, $Z_n \leftarrow Z_n + 1$
31 **end**
32 $L \leftarrow \emptyset$
33 **end**
34 $\Delta_n \leftarrow 1/|B_n|$
35 **for** $f_i \in F'$ **do**
36 $r_i \leftarrow r_i + \Delta_n$
37 **end**
38 $F \leftarrow F'$, $F' \leftarrow \emptyset$
39 **end**
40 **end**
41 $S \leftarrow B_{n-1}$, Return S and r

Proposition 1: For each iteration n, Algo-Fair generates a superframe B_n with a finite length of $|B_n| = Z_n \times |B_{n-1}|$, where $Z_n \leq |\mathcal{L}| \leq |E||\mathcal{F}|$.

Proof: Recall that $\mathcal{L}$ is a multiset containing all links belonging to opportunistic flows. As there are a finite number of links and flows, the size of $|\mathcal{L}|$ is less than or equal to $|E| \times |\mathcal{F}|$. Hence, the size of multiset $|\mathcal{L}|$ is finite. Our scheduler, see Algorithm 1, moves to the next iteration only if set $\mathcal{L}$ is empty. This means all links in $\mathcal{L}$ will be served in each iteration n. To see this guarantee, consider *line 21-35*. Every time Algo-Fair augments the superframe in iteration n, at least one link from $\mathcal{L}$ is scheduled in the newly copied superframe B_{n-1}. Therefore, in each iteration, the number of augmentations is finite and there are only a maximum of $|\mathcal{L}|$ such operations. ■

As a result of Proposition 1, the value of Z_n is upper bounded by $|E||\mathcal{F}|$. This implies there are a maximum $|E||\mathcal{F}|$ copies of B_{n-1} in iteration n. Next, we show that the number of iterations is finite. We first make the following definition.

Definition 7: All links that are assigned a slot in the last copied B_{n-1} of superframe B_n are called 'the last links'.

Proposition 2: A last link l that is assigned slot k in iteration n will never be assigned the same slot k again in any later iterations.

Proof: In iteration n, if a link l is one of the last links, then all its available opportunistic slots before the last copied B_{n-1} superframe must be occupied by conflicting links. Otherwise, link l would have been scheduled earlier; i.e., it is not a last link. In order to schedule link l, Algo-Fair augments the superframe with a copy of B_{n-1}, meaning at least one opportunistic slot of link l is available. Otherwise, it would not have been marked an opportunistic link in iteration n. Assign link l one of these slots, say slot k. Consequently, in every copy of B_n used to construct B_{n+1}, slot k is occupied by link l. This implies in all subsequent iterations, this slot will also be occupied. This proves the statement. ■

As each iteration has at least one last link, all links are guaranteed to exhaust their opportunistic slots, and thereby, at such time, they cannot be activated opportunistically. This fact leads to the next proposition.

Proposition 3: Algo-Fair will terminate after a finite number of iterations.

Proof: In every iteration n, there must be $x \geq 1$ last links, meaning there will be x links that lose at least one available opportunistic slot because the value of Z_n is finite, as per Proposition 1, where $1 \leq x \leq |\mathcal{L}|$. Hence, after every iteration, at least one link in $\mathcal{L}$ will lose at least one opportunistic slot. In iteration n, each link has a finite number of opportunistic slots because the number of superframes in B_{n-1} and the value of Z_n are both finite. Thus, every link will lose all its opportunistic slots after a finite number of iterations. Once there are no opportunistic slots, the set F' is empty, and Algo-Fair terminates. ■

As mentioned in Section IV, we use the stopping criterion ϵ, where its main functionality is to reduce the computation time and to ignore negligible increase in flow rate as Z_n becomes large. Observe that n may be very large. Assume a link l in $\mathcal{L}$ has m opportunistic slots in iteration n. According to Proposition 1, the maximum number of new opportunistic slots is $|\mathcal{L}| \times (m-1)$ in iteration $n+1$. Thus, we need at least

$|\mathcal{L}| \times (m-1)$ iterations to exhaust all of link l's opportunistic slots after iteration n.

Recall that Δ_n is the additional bandwidth, due opportunistic slots, allocated to a flow in each iteration, where Δ_n is equal to $\frac{1}{|B_n|}$. When the value of n is large, Δ_n is very close to zero. Thus, any flow rate increase is negligible when $0 < \Delta_n < \epsilon$, and thus it is save to terminate.

To prove that our scheduler approximates MMF, we make the following two propositions. Recall that one definition of MMF is that the rate of a flow cannot be increased without decreasing the rate of other flows with less or equal rates [3]. Also note that Algo-Fair allocates opportunistic flows the same flow rate increment in each iteration without sacrificing the rate of existing flows.

Proposition 4: The rate allocation produced by Algo-Fair in iteration n is never less than that in iteration $n-1$.

Proof: Assume link l is active in slot k in superframe B_{n-1} of iteration $n-1$. In the (n)-th iteration, the new superframe B_n includes one or more copies of B_{n-1}. Observe that existing allocations are preserved. This is evident in Figure 4. For example, the links in B_0 are not affected in each iteration. In particular, if link l is active in slot k, then its next activation will be in $k + |B_{n-1}|$, $k + |B_{n-1}| \times 2, \ldots, k + |B_{n-1}| \times Z_{n-1}$. In other words, the frequency of transmission is the same. If a link is activated opportunistically, by the definition of opportunistic links, see Section III, it does not increase the superframe length, and thus the rate allocation can only increase. ■

To satisfy another aspect of MMF, we need to show that every flow has at least one bottleneck link.

Proposition 5: Every flow $f_i \notin F'$ *has at least one bottleneck link.*

Proof: A link l of a flow f_i is a bottleneck if and only if link l's capacity is consumed by all flows crossing it and the rate of f_i is higher than or equal to the rate of other flows crossing l. We first prove that link l is saturated. Recall that the number of slots allocated to a link corresponds to its capacity. Consider the case in which flow f_i was an opportunistic flow in iteration $n-1$ but is no longer one in iteration n; i.e., it is not in the set F'. By the definition of F', flow f_i has at least one link that does not have an opportunistic slot in B_{n-1}; assume this to be link l. This means no more slots can be allocated to link l to increase its capacity; i.e., it is saturated. Secondly, we prove that the rate of f_i is higher than or equal to the rate of other flows crossing l. Consider the case where flow f_i is not the only flow crossing link l. Let there be z flows that were in F' in iteration $n-1$ but are no longer in F' in iteration n. Referring to *line 37-39* in Algorithm 1, all flows in F' will receive the same additional bandwidth in iteration 1 to $n-1$. This means all z flows and f_i will be allocated the same bandwidth up to iteration $n-1$. Then the maximum rate allocated to these z flows is $1/|B_0| + 1/|B_1| + \cdots + 1/|B_{n-1}|$. Thus, the rate of f_i is higher than or equal to the rate of other flows crossing l, meaning l is a bottleneck link. ■

Finally, we have the following time complexity result.

Proposition 6: Algo-Fair has a time complexity of $O(|E| \times |\mathcal{F}| \times |V|^2)$.

Proof: Consider Algorithm 1. The time complexity of line 1 and 3 is $O(|\mathcal{F}|)$. Line 2 needs to run Algo-2, which has a time complexity is $O(|V|^2)$. Thus, the time complexity of line 1-4 is $O(|\mathcal{F}|) + O(|V|^2) + O(|\mathcal{F}|) + O(1) + O(1) + O(1) + O(1)$. Assuming $|\mathcal{F}| < |V|^2$, we thus have $O(|V|^2)$. Lines 9-11 need to check if each link of flow f_i is an opportunistic link in all slots of B_{n-1}. Thus, the complexity of lines 9-11 is $O(|E| \times |B_{n-1}|)$. The time complexity of lines 13-14 is $O(|E|)$. The iteration from lines 7-16 needs to check all flows in F. Thus, the time complexity is $O(|F| \times |E| \times |B_{n-1}|)$, where $|B_{n-1}| = |B_0| \times Z_1 \times Z_2 \times \cdots \times Z_{n-1}$. The time complexity of lines 17-20 is $O(1)$. Lines 22-24 need to check each link in $|\mathcal{L}|$, where $|\mathcal{L}| \leq |E| \times |\mathcal{F}|$. The complexity of lines 25-28 is $O(|V|^2)$ because the MAXCUT() function uses Algo-2. Lines 29-34 has complexity $O(1)$. The total time complexity of lines 22-34 is $O(|E| \times |\mathcal{F}|) + O(|V|^2) + O(1)$. If the value of $|\mathcal{F}|$ is less than $\frac{|V|^2}{|E|}$, then it becomes $O(|V|^2)$. The maximum number of iterations from lines 21 to 35 is $|\mathcal{L}|$. Thus, the time complexity of line 21-35 is $O(|E| \times |\mathcal{F}| \times |V|^2)$. The time complexity of line 36-40 is $|F'|$. Thus the total time complexity of Algorithm 1 is $O(|F| \times |E| \times |B_{n-1}|) + O(1) + O(|E| \times |\mathcal{F}| \times |V|^2) + O(|F'|)$. If $|B_{n-1}| < |V|^2$, then the time complexity of our MMF scheduling algorithm is $O(|E| \times |\mathcal{F}| \times |V|^2)$. ■

VII. EVALUATION

Our experiments are conducted using Matlab with the MatGraph toolkit [22]. We assume all nodes are stationary and located randomly on a 200×200 m^2 or 800×800 m^2 square area. The number of nodes ranges from 10 to 150. The transmission range is set to 100 or 400 meter depending on the experiment. We assume each node has a dedicated antenna/beam for each neighbor, all nodes operate on the same frequency, and each link l operates at a maximum rate of C_l =10 Mbit/s. The number of flows $|\mathcal{F}|$ ranges from 5 to 75. For each flow, we randomly select a source and destination node, and route it over the shortest path. We set $\epsilon = 0.001$; equivalent to when the number of slots in $|B_n|$ is more than 1000. We note that Algo-Fair rarely reaches this ϵ limit in our experiments, where the additional bandwidth provided to flows is negligible. Each result point is an average of 10 experimental runs on the same topology. We emphasize that our focus is on superframe construction and slot allocation. In other words, we are not concerned with channel conditions. Having said that, we note that it is possible to compute the expected flow rate in non-ideal channel conditions. This is because Algo-Fair assigns one or more time slots to links that repeat periodically, and from that we can determine the link capacity in perfect channel conditions. If the channel is not perfect, then we can scale the flow rate according to a given bit error rate. That is one can determine the expected number of failures or slots required to transmit a packet successfully.

We evaluate Algo-Fair against six other designs, namely, Algo-2OL, Algo-1OL, Algo-2Flow, Algo-1Flow, A2Greedy and A1Greedy. The main difference among them is as follows. Algo-2OL, Algo-2Flow and A2Greedy apply Algo-2 [11], and Algo-1OL, Algo-1Flow and A1Greedy use Algo-1 [5]. To aid readability, we use the label '1' or '2' to denote the scheduler in question, e.g., Algo-2OL and Algo-1OL correspond to Algo-2 [11] and Algo-1 [5], respectively. In our experiments, they serve as example schedulers that do not consider flow fairness. Note that Algo-2Flow and Algo-1Flow emulate the behavior of the end-to-end water filling algorithm of [20]. A key difference, apart from being distributed, is that the algorithm in [20] assumes a tree topology whose nodes form a matching in each time slot, i.e., one transmission/reception per node.

We now briefly explain the six schedulers. Algo-2OL and Algo-1OL first generate a basic superframe. After that, both algorithms add all possible opportunistic links into each slot. If all links of a flow are added as opportunistic links, then its rate will increase. In contrast, after generating a basic superframe, Algo-2Flow and Algo-1Flow increase the weight of flows iteratively. Specifically, they randomly pick a flow, and increase its weight by one. A new superframe is then generated. If the resulting superframe causes other flows' rate to reduce, they then revert the flow's weight to its previous value. Otherwise, they retain the flow's weight and move to the next flow. This process repeats until no flow weight can be increased. The last two algorithms, A2Greedy and A1Greedy, focus on throughput rather than fairness. They iteratively consider flows with increasing hop count, starting from the shortest, and generating a new superframe in each iteration to accommodate newly added flows. If a flow, say f_z, causes the rate of other flows to reduce, they revert to the superframe of the previous iteration, and f_z is removed from consideration.

In each experiment, we collected the following metrics:

- *Flow rates*. The final flow rate is equal to the minimum link rate along its path. This is computed using Equ. (1), where n_i^l is the number of slots assigned to link l for flow f_i, and $|S|$ is the superframe length,

$$r_i = MIN\left\{\frac{n_i^l}{|S|} \mid l \in f_i\right\} \tag{1}$$

- *Average total throughput*. This is simply the sum of all flow rates r_i divided by the number of simulation runs, which is 10.
- *Average end to-end delay*. This is the average time a packet takes to make its way from a source to a destination.

The first experiment is over three $200 \times 200\ m^2$ randomly generated topologies with 10, 15 and 20 nodes. In Figure 6, the minimum flow rate generated by Algo-Fair is 1.67 Mbit/s when the number of nodes is 10. This is higher than the minimum rate generated by Algo-1OL, Algo-1Flow, A1Greedy, i.e., 1.25 Mbit/s, and A2Greedy, i.e., zero. The reason is

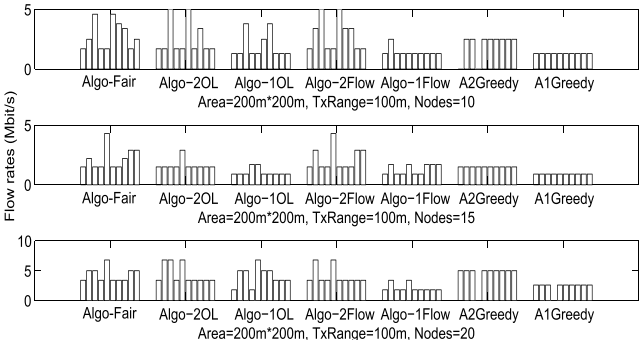


FIGURE 6. Allocated rates of ten flows on three topologies located in an $200 \times 200\ m^2$ area with 10, 15 and 20 nodes.

because Algo-1 generates longer basic superframe lengths than Algo-2, and the minimum flow rate is decided by the basic superframe. A2Greedy starved two flows while allocating other flows with a rate of 2.5 Mbit/s. This is because A2Greedy assigns a higher priority to shorter flows. When the number of nodes is 15, the minimum rate generated by Algo-Fair is 1.43 Mbit/s. This is higher than the minimum rate, i.e., 0.83 Mbit/s, generated by Algo-1OL, Algo-1Flow and A1Greedy. Also observe that Algo-Fair equally allocates an additional 1.43 Mbit/s to two flows rather than allocating 1.43 Mbit/s to one flow as is the case with Algo-2Flow. When the number of nodes is 20, A2Greedy and A1Greedy starved one flow. Algo-1OL and Algo-1Flow ensure a minimum rate of 1.67 Mbit/s. This is lower than the minimum rate, i.e., 3.33 Mbit/s, generated by Algo-Fair because Algo-1 generates longer superframes than Algo-2. In this case, Algo-Fair equally allocates an additional 6.67 Mbit/s to four flows rather than allocating 6.67 Mbit/s to two flows. In contrast, Algo-2OL allocates an additional 6.67 Mbit/s to two flows and Algo-2Flow allocates 3.33 Mbit/s to one flow.

We also compare the flow rates computed by all algorithms to the optimal MMF rate. This is carried out manually, and hence, is only possible for small topologies. Consider Figure 5. We assign all flows a weight of one, meaning each link's weight is equal to the number of traversing flows. Then we set the weight of a node to the sum of the maximum incoming link weight and the maximum outgoing link weight. In Figure 5, node E is the heaviest node because it has an incoming link with a weight of two and an outgoing link with a weight of two. We thus mark node E as a bottleneck node, and link DE and EF are bottleneck links. In order to satisfy the no Mix-Tx-Rx constraint, each flow traversing link DE and EF can only receive a fair share of 1/4. Thus, the rate of flow f_1, f_3 and f_4 is 2.5 Mbit/s. We see that flows f_2 and f_1 share link BC. Flow f_1 is bottlenecked by link DE and EF. Thus, flow f_2 can use the remaining capacity of link BC, which is $1-1/4 = 3/4$. The flow f_2 will have a rate of $3/4 \times 1/2 = 3/8$ because node B and C need to receive and transmit data for flow f_2. Thus, the rate of flow f_2 is 3.75 Mbit/s. Recall that earlier in Section V, using our Algo-Fair, we computed the following rates: $r_1 = 1/4$, $r_2 = 3/8$, $r_3 = 1/4$ and $r_4 = 1/4$. These rates are in fact optimal.

TABLE 5. Average gaps as compared to the optimal MMF.

Algorithm	Average Gap		
	Nodes=10	Nodes=15	Nodes=20
Algo-Fair	0.21	0.14	0
Algo-2OL	0.5	0.43	0.67
Algo-1OL	1.13	0.75	1.17
Algo-2Flow	0.33	0.21	0.67
Algo-1FLow	1.63	0.33	1.17
A2Greedy	1.33	0.43	0.5
A1Greedy	1.75	0.67	1.25

Table 5 shows the gap as compared to the optimal flow rates. We see that Algo-Fair has the lowest average gap, i.e., 0.21 Mbit/s and 0.14 Mbit/s, for the topology with 10 nodes and 15 nodes respectively. When the number of nodes is 20, Algo-Fair matches the optimal rates. The gap between the results of Algo-Fair and the optimal flow rates is due to two reasons. First, we note that Algo-2 is a heuristic for the NP-complete, MAXCUT problem. Thus, it may not generate the shortest superframe for a given topology. In other words, the minimum rate received by flows is non-optimal. Second, as opportunistic links are added greedily, see *line 26* of Algorithm 1, the resulting allocation of opportunistic links may cause unnecessary augmentations.

We now study increasing number of flows, i.e., 5 to 75, using a randomly generated topology on a $800 \times 800\ m^2$ area with 75 nodes. From Figure 7, we see that Algo-Fair generates the highest throughput, with an average throughput that is 8.3 Mbit/s higher than Algo-2Flow, and 130.5 Mbit/s higher than A1Greedy. When there are five flows, Algo-Fair, Algo-2OL and Algo-2Flow are both at 36 Mbit/s because their first two steps are the same. When there are 60 flows, Algo-Fair increases the average total throughput by more than 73.2% as compared to A2Greedy, and about 10.4% as compared to Algo-2OL and about 6.9% as compared to Algo-2Flow. When using A2Greedy, longer flows may starve. In Algo-2OL, the rate allocated to a link will increase significantly if this link is assigned several opportunistic slots. However, the rate of a flow is constrained by the minimum link rate along its path. The results generated by Algo-2Flow are very close to those of Algo-Fair when the number of flows is small, i.e., 5, 10, 15 and 20. Nevertheless, if two opportunistic flows, say f_i and f_j, share one link l, and l only has one opportunistic slot in the generated superframe, Algo-2Flow may not fairly increase the rates of the two flows. If Algo-2Flow first selects flow f_i and increases its weight by one, then flow f_j's weight cannot be increased without adding a new slot to link l. The rate allocated to flows will decrease if the superframe length increases. Thus, Algo-2Flow cannot fairly allocate additional bandwidth in some cases.

We now study the effect of node numbers and degrees on the average total throughput. The number of flows is fixed at 30. Both Figure 8 and 9 show that Algo-Fair generates the highest throughput. With varying node numbers, on average, the throughput of Algo-Fair is 11 Mbit/s and 102.6 Mbit/s higher than the throughput of Algo-2Flow and

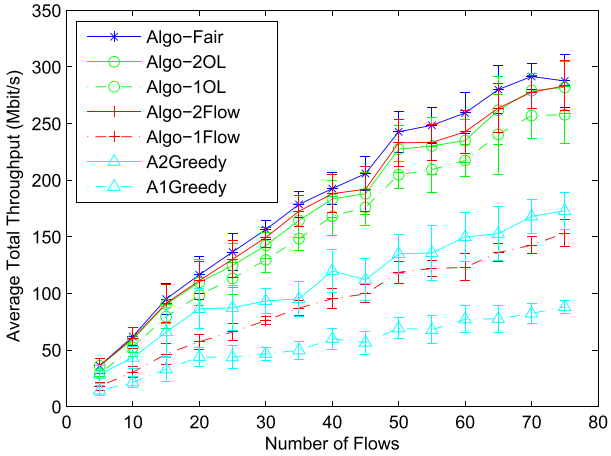


FIGURE 7. Average total throughput with different number of flows.

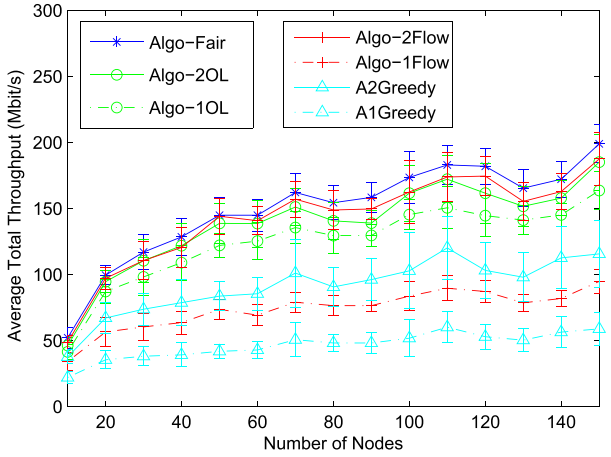


FIGURE 8. Average total throughput with different number of nodes.

A1Greedy, respectively. As for different node degrees, Algo-Fair recorded an average that is 4.1 Mbit/s and 48.2 Mbit/s higher than Algo-2Flow and A1Greedy, respectively. In both experiments, when there is a higher number of nodes or node degrees, there are more links. Consequently we see a drop in the number of flows traversing each link. In contrast, the load on each link is higher when the number of nodes or node degrees is small. In this scenario, Algo-Fair will assign the same additional bandwidth to each contending flow. However, in the other six algorithms, one or more flows are starved. On the other hand, in the former case, as there are fewer flows on each link, they will receive a higher rate. We see that Algo-Fair also performs well as it maximizes the rate of each flow while the other six algorithms only maximize the rates of a subset of flows.

Lastly, we study end-to-end delays with increasing number of flows. The source and destination of each flow are assigned randomly. Figure 10 shows the average end-to-end delay for the tested designs. We see that Algo-Fair has almost similar performance to Algo-2OL. Algo-2OL yields slightly smaller delays because some links receive more opportunistic slots.

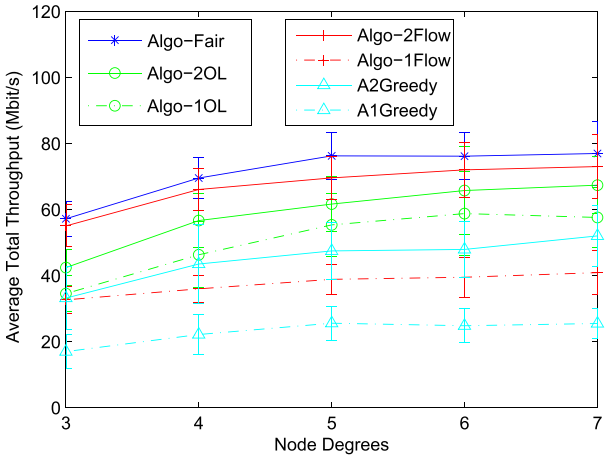


FIGURE 9. Average total throughput with different node degrees.

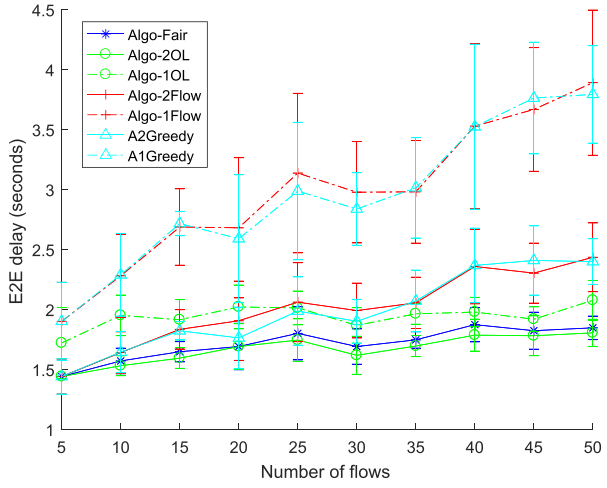


FIGURE 10. Average end-to-end (E2E) delay with increasing number of flows.

In contrast, for Algo-Fair, opportunistic slots are shared fairly amongst links to ensure the corresponding flows using those links receive a fair share of the additional link capacity. As for Algo-1OL, the average end-to-end delay is higher than Algo-Fair because the resulting superframe length is longer; please see [11] for a comparison between the superframe lengths generated by Algo-1 and Algo-2. This explains the larger end-to-end delays for schedulers that are based on Algo-1. Both Algo-2Flow and A2Greedy optimize the superframe for certain flows only; e.g., A2Greedy prefers short flows. Consequently, some flows, e.g., those over longer routes when using A2Greedy, experience higher delays as their links are not activated sufficiently often in the resulting superframe.

VIII. CONCLUSION

In summary, Algo-Fair has the following advantages: 1) it ensures all flows are allocated a minimum rate, meaning no flows starve, 2) it uses Algo-2 to ensure the highest possible minimum rate, and 3) it employs an augmentation operation to ensure all opportunistic flows are assigned a fair increase in rate. The key novelty is step 3) where a novel augmentation step is used to exploit opportunistic slots. We believe a similar step can be applied when deriving the superframe for other wireless systems. Compared with other designs, such as Algo-2OL, Algo-1OL, Algo-2Flow, Algo-1Flow, A2Greedy and A1Greedy, experiment results show Algo-Fair yields higher fairness in all tested scenarios. Moreover, Algo-Fair and Algo-2OL yield the lowest end-to-end delays.

Lastly, recall that the network considered is general and applies to three different MTR systems. To this end, a key future work will be to consider physical layer characteristics of a MTR system; e.g., Degree of Freedoms (DoFs) assignment in a MIMO-based MTR WMN to optimize the number of transmitting or receiving links on each node, minimize interference and thus increase data rates and/or create full-duplex links using the method in [32]. Additionally, a possible future work will be to incorporate algorithms such as those in [30] to minimize the end-to-end delay of flows by re-ordering slots. Another future work will be to apply our augmentation step to spatial TDMA, non-MTR based wireless systems.

REFERENCES

[1] S. Bai, W. Zhang, Y. Liu, and C. Wang, "Max-min fair scheduling in OFDMA-based multi-hop WiMAX mesh networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kyoto, Japan, Jun. 2011, pp. 1–5.

[2] C. Bernardos *et al.*, "An architecture for software defined wireless networking," *IEEE Wireless Commun.*, vol. 21, no. 3, pp. 52–61, Jun. 2014.

[3] D. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1992.

[4] M. Cheng and Q. Ye, "A combinatorial solution for scheduling spatial multiplexing in MIMO-based ad hoc networks," in *Proc. IEEE GLOBECOM Workshops*, Anaheim, CA, USA, Dec. 2012, pp. 931–936.

[5] K.-W. Chin, S. Soh, and C. Meng, "Novel scheduling algorithms for concurrent transmit/receive wireless mesh networks," *Comput. Netw.*, vol. 56, no. 4, pp. 1200–1214, Mar. 2012.

[6] S. Chu and X. Wang, "Opportunistic and cooperative spatial multiplexing in MIMO ad hoc networks," *IEEE/ACM Trans. Network.*, vol. 18, no. 5, pp. 1610–1623, Oct. 2010.

[7] G. R. Hiertz *et al.*, "IEEE 802.11s: The WLAN mesh standard," *IEEE Wireless Commun.*, vol. 17, no. 1, pp. 104–111, Feb. 2010.

[8] M. Kas, B. Yargicoglu, I. Korpeoglu, and E. Karasan, "A survey on scheduling in IEEE 802.16 mesh mode," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 2, pp. 205–221, 2nd Quart., 2010.

[9] B. Li, "End-to-end fair bandwidth allocation in multi-hop wireless ad hoc networks," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Columbus, OH, USA, Jun. 2005, pp. 471–480.

[10] Y. Liu, Y. Xiong, Y. Yang, P. Xu, and Q. Zhang, "An experimental study on multi-channel multi-radio multi-hop wireless networks," in *Proc. IEEE GLOBECOM*, St. Louis, MO, USA, Nov. 2005, p. 5.

[11] H. Loo, S. Soh, and K.-W. Chin, "On improving capacity and delay in multi Tx/Rx wireless mesh networks with weighted links," in *Proc. IEEE APCC*, Bali, Indonesia, Aug. 2013, pp. 12–17.

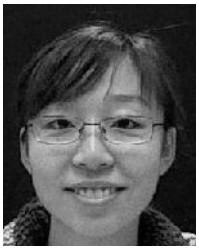
[12] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, Baltimore, MD, USA, Nov. 2004, pp. 39–49.

[13] R. Mudumbai, S. Singh, and U. Madhow, "Medium access control for 60 GHz outdoor mesh networks with highly directional links," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009, pp. 2871–2875.

[14] B. Mumey, J. Tang, and T. Hahn, "Joint stream control and scheduling in multihop wireless networks with MIMO links," in *Proc. IEEE ICC*, Beijing, China, May 2008, pp. 2921–2925.

[15] Y. Niu, Y. Li, D. Jin, L. Su, and D. O. Wu, "A two stage approach for channel transmission rate aware scheduling in directional mmWave WPANs," *Wireless Commun. Mobile Comput.*, vol. 16, no. 3, pp. 313–329, 2014.

[16] A. Penttinen, I. Koutsopoulos, and L. Tassiulas, "Low-complexity distributed fair scheduling for wireless multi-hop networks," in *Proc. 1st Workshop Resour. Allocation Wireless Netw.*, Riva Del Garda, Italy, Apr. 2005.
[17] M. Pióro, M. Żotkiewicz, B. Staehle, D. Staehle, and D. Yuan, "On max–min fair flow optimization in wireless mesh networks," *Ad Hoc Netw.*, vol. 13, pp. 134–152, Feb. 2014.
[18] J. Qiao, L. X. Cai, X. Shen, and J. W. Mark, "Stdma-based scheduling algorithm for concurrent transmissions in directional millimeter wave networks," in *Proc. IEEE ICC*, Ottawa, ON, Canada, Jun. 2012, pp. 5221–5225.
[19] B. Raman and K. Chebrolu, "Design and evaluation of a new mac protocol for long-distance 802.11 mesh networks," in *Proc. ACM MOBICOM*, Cologne, Germany, Aug. 2005, pp. 156–169.
[20] T. Salonidis and L. Tassiulas, "Distributed dynamic scheduling for end-to-end rate guarantees in wireless ad hoc networks," in *Proc. 6th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, Urbana-Champaign, IL, USA, May 2005, pp. 145–156.
[21] S. Sarkar and L. Tassiulas, "End-to-end bandwidth guarantees through fair local spectrum share in wireless ad-hoc networks," *IEEE Trans. Autom. Control*, vol. 50, no. 9, pp. 1246–1259, Sep. 2005.
[22] E. R. Scheinerman, "Matgraph: A MATLAB toolbox for graph theory," Ph.D. dissertation, Dept. Appl. Math. Statist., The Johns Hopkins Univ., Baltimore, MD, USA, 2008.
[23] H. Shi, R. V. Prasad, E. Onur, and I. Niemegeers, "Fairness in wireless networks:Issues, measures and challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 5–24, 1st Quart., 2014.
[24] J. Song and K.-W. Chin, "A survey of single and multi-hop schedulers for mmWave wireless networks," *Elsevier Ad Hoc Netw.*, vol. 33, pp. 269–283, Oct. 2015.
[25] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.
[26] L. Tassiulas and S. Sarkar, "Maxmin fair scheduling in wireless ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 163–173, Jan. 2005.
[27] V. V. Vazirani, *Approximation Algorithms*. Berlin, Germany: Springer-Verlag, 2004.
[28] H. Wang, K.-W. Chin, R. Raad, and S. Soh, "A distributed maximal link scheduler for multi Tx/Rx wireless mesh networks," in *Proc. IEEE ICC*, Sydney, NSW, Australia, Jun. 2014, pp. 2779–2784.
[29] H. Wang and W. Jia, "Design a novel fairness model in WiMAX mesh networks," *Comput. Commun.*, vol. 35, no. 12, pp. 1447–1456, 2012.
[30] L. Wang, K.-W. Chin, R. Raad, and S. Soh, "Delay aware joint routing and scheduling for multi-Tx-Rx wireless mesh networks," in *Proc. IEEE ICC*, Sydney, NSW, Australia, Jun. 2014, pp. 2773–2778.
[31] P. Wang, H. Jiang, W. Zhuang, and H. V. Poor, "Redefinition of max-min fairness in multi-hop wireless networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 12, pp. 4786–4791, Dec. 2008.
[32] X. Xie and X. Zhang, "Semi-synchronous channel access for full-duplex wireless networks," in *Proc. IEEE ICNP*, Raleigh, NC, USA, Oct. 2014, pp. 1–8.
[33] Y. Xu, K.-W. Chin, S. Soh, and R. Raad, "A max weight distributed scheduler for multi Tx/Rx wireless mesh networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 11, pp. 9345–9357, Nov. 2016.



LUYAO WANG received the B.Eng. degree (Hons1) from Zhengzhou University and University of Wollongong, and the Ph.D. degree from the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong. Her research interests are scheduling algorithms for wireless mesh networks.

KWAN-WU CHIN received the B.Sc. (Hons1) and Ph.D. degrees from the Curtin University of Technology, Australia, where he graduated with the Vice-Chancellor's Commendation. He is an Associate Professor with the University of Wollongong. He joined Motorola Research Lab as a Senior Research Engineer, where he developed zero configuration home networking protocols and designed new medium access control protocols for wireless sensor networks and next-generation bandwidth managers. He has filed nine U.S. patents and received a major grant from DARPA. In 2004, he joined the University of Wollongong as a Senior Lecturer before being promoted to an Associate Professor in 2011. He is also the Head of Post-Graduate Studies for the School of Electrical, Computer and Telecommunications Engineering and the Director of the Wireless Technologies Lab. His current research areas include medium access control protocols for wireless networks, resource allocation algorithms/policies for communications networks, routing protocols for delay tolerant networks, and mathematical programming. To date, he holds four U.S. patents and has published more than 100 conference and journal articles.

SIETENG SOH (M'11) received the B.S. degree in electrical engineering from the University of Wisconsin-Madison, Madison, WI, USA, and the M.S. and Ph.D. degrees in electrical engineering from Louisiana State University, Baton Rouge, LA, USA. From 1993 to 2000, he was a Faculty Member with Tarumanagara University, Indonesia, where he was the Director of the Research Institute from 1998 to 2000. He is currently a Lecturer with the Department of Computing, Curtin University, Perth, Western Australia. His research interests include network reliability, and parallel and distributed processing. He is a member of the IEEE Computer Society.

TENGJIAO HE received the B.Eng. degrees (Hons.) in computer engineering from the University of Wollongong, Australia, and Zhengzhou University, China, and the Ph.D. degree from the University of Wollongong in 2016. His research interest is resource management issues in wireless networks.